This is the manuscript of a paper given at the International Workshop on Data Analysis in Astronomy, held at Erice (Italy) from 28 May to 4 June 1984.
The proceedings of this workshop have been edited by V. di Gesu and L. Scarsi and have been published by Plenum Press (New York, London) in 1985, pp 271-302.

THE GRONINGEN IMAGE PROCESSING SYSTEM

R.J. Allen, R.D. Ekers*, J.P. Terlouw

Kapteyn Astronomical Institute
University of Groningen
Postbus 800, 9700 AV GRONINGEN, The Netherlands

"The obsolescence of an implementation must be measured against other existing implementations, not against unrealized concepts".

Brooks, 1975; p.9.

SUMMARY

An interactive, integrated software and hardware computer system for the reduction and analysis of astronomical images is described. After a short historical introduction, some examples of the astronomical data currently handled by the system are shown, followed by a description of the present hardware and software structure. The system is then further illustrated by describing its appearance to the user, to the applications programmer, and to the system manager. Finally, some quantitative information on the size and cost of the system is given, and its good and bad features are discussed.

*Present address: National Radio Astronomy Observatory, P.O. Box 0, SOCORRO, N.M. 87801, U.S.A.

I. INTRODUCTION

In this paper we describe the current state of an image processing system which has been under continuous development since 1971. This system was originally installed as a single interactive program in a PDP-9 computer with a single-user operating system and modest hardware resources, as described previously by Ekers, Allen, and Luyten (1973). The next version ran as a single program in a time-sharing CDC 6600 computer during the period 1976-78. The present version, commonly known by the acronym GIPSY, is an organized set of independent, interactive programs which runs in a PDP 11/70 computer with advanced image display facilities and a multi-user, multi-tasking operating system.

The design and development of GIPSY has been strongly driven by the needs of its users, and it is the users themselves who continue to add most of the new software to the system. Details of the investment in manpower and of the current size of the system are given later in this paper; a summary can also be found in the paper by Allen and Ekers elsewhere in this volume.

Many aspects of GIPSY software architecture have influenced the designers of newer systems. The original STARLINK project managers decided to include a number of GIPSY features into the definition of their first-generation "software environment" in 1980 (SGP /7). The major new element in their design was to take specific advantage of the facilities offered by the VMS operating system in the VAX computers used on STARLINK. Although this plan was largely abandoned in 1981 in favour of a new scheme (SGP/18.1), several other groups had already begun the design of systems based on the first-generation STARLINK software environment. In this indirect way, GIPSY has had an effect on MIDAS and FIPS (both described in this volume). Also, the development of PANDORA (Simkin, Bosma, Pickles, Quinn, and Warne 1983) at Mount Stromlo in Australia grew out of an early version of GIPSY installed there in 1979.

At the start in 1971, application programs written by the users for GIPSY concentrated on the analysis of two-dimensional radio astronomy map data obtained from the Westerbork Synthesis Radio Telescope (WSRT). With the advent of spectroscopy at the WSRT, the architecture and application programs were expanded beginning around 1976 in order to allow processing three-dimensional data. Presently, the users of GIPSY analyse observations made with a variety of radio, infrared, and optical detection systems. Some examples of the kinds of data currently being processed are given in section II.

Our description of GIPSY will follow the general scheme suggested by Allen elsewhere in this volume. Section III describes the current hardware and software structure of the system. Section IV

discusses the appearance of GIPSY to the user, to the programmer, and to the system manager. Finally, in section V we present some statistics about the system and comment on its good and bad features.

II. SOME EXAMPLES OF ASTRONOMICAL DATA

In our paper describing the first generation system (Ekers et al. 1973), we showed a number of examples of the radio continuum aperture sythesis data which was being analysed at that time. Six years later, multiple-image radio astronomy data sets were being accomodated in GIPSY, and methods of displaying such enormous quantities of data were being explored (Allen, 1979b). Presently, multiple-image optical data cubes and infrared observations are also routinely analysed.

We have chosen a few recent examples from current research programs; for the most part these results have not yet been shown elsewhere. Figure 1 illustrates a multiple-image optical data cube, and Figure 2 shows the results of some analysis on that data. Figures 3 and 4 have been obtained by a similar analysis of a radio data cube. Finally, Figures 5 and 6 show exaples of radio and infrared continuum maps.

III. IMPLEMENTATIONS

1. Hardware Configuration

The configuration of computer hardware in which the present version of GIPSY is implemented in Groningen is shown in Figure 7. The host computer and its standard peripherals are provided by the Groningen University Computer Center as a general facility to the university research community. Astronomical image processing is the major activity, but the computer is also used by several other groups. For image processing and display, three additional devices are also connected as peripherals to the host computer. The function of these devices is described below.

Image Computer and Display. This consists of a commercially-available device[a] to which a few minor local modifications have been made. It is used for the manipulation and display of digital images in a variety of sizes and data formats. The internal organization consists of six image memories of 512 × 512 8-bit bytes, but these memories can be re-configured for multiple-byte words and for larger or smaller images. Besides the standard display (e.g.

---

a. Model 70E, International Imaging Systems, 1500 Buckeye Drive, Milpitas, CA 95035.

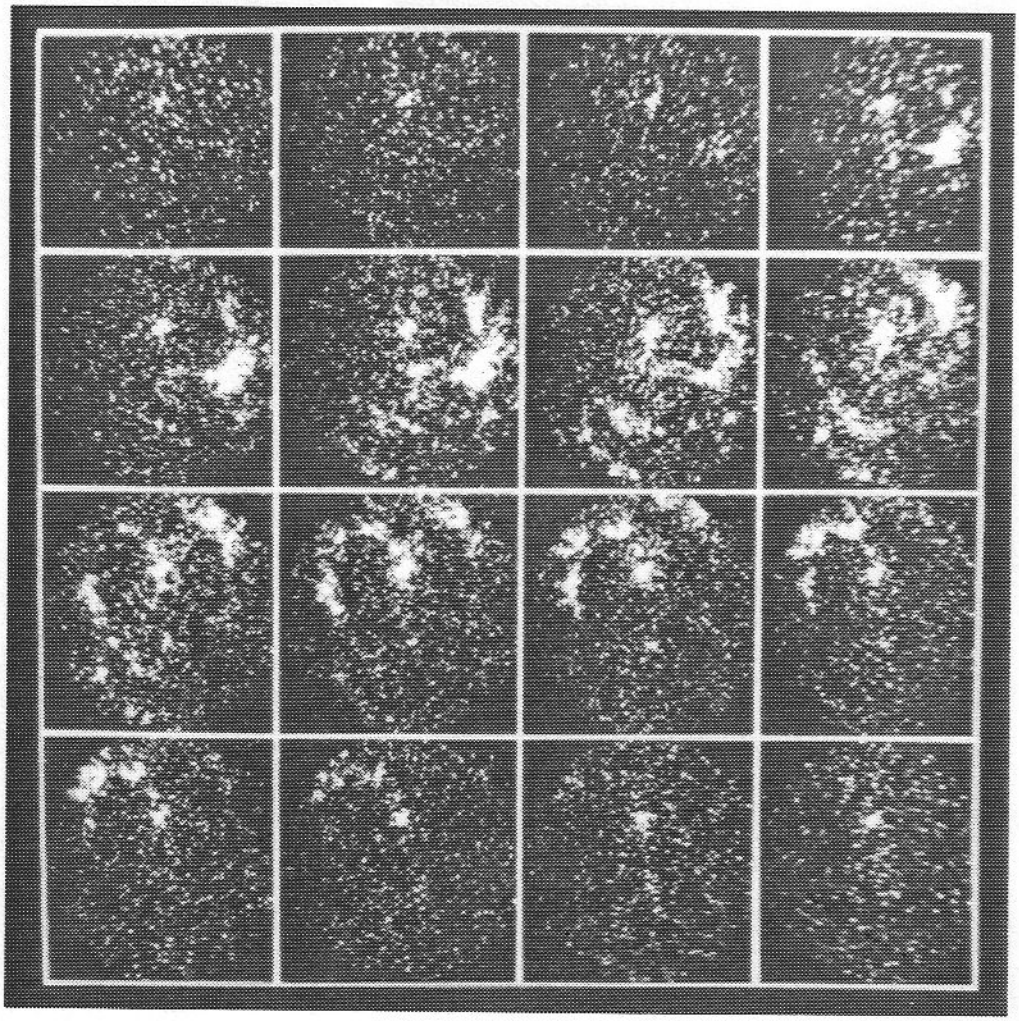

Fig. 1 Portions of a seeing-limited data cube of optical Hβ line emission from a central 5.5 arc minute field on the southern spiral galaxy M83 = NGC 5236, as observed with the TAURUS imaging Fabry-Perot spectrometer and the Image Photon Counting System detector on the Anglo-Australian Telescope. The full data cube is 256 × 256 pixels × 98 channels. The channels selected for this mosaic have a velocity width of about 27 km s^{-1} and are separated by 20.7 km s^{-1}, beginning at 670 km s^{-1} in the upper left corner, and decreasing to 360 km s^{-1} at the lower right. The continuum emission has been estimated from channels outside the range of the Hβ line and subtracted from each remaining channel; however, owing partly to intrumental effects, the bright nucleus of the galaxy remains visible just to the north of the center on each channel map. From unpublished work by R.P.J. Tilanus; see also Allen, Atherton, and Tilanus (1985).

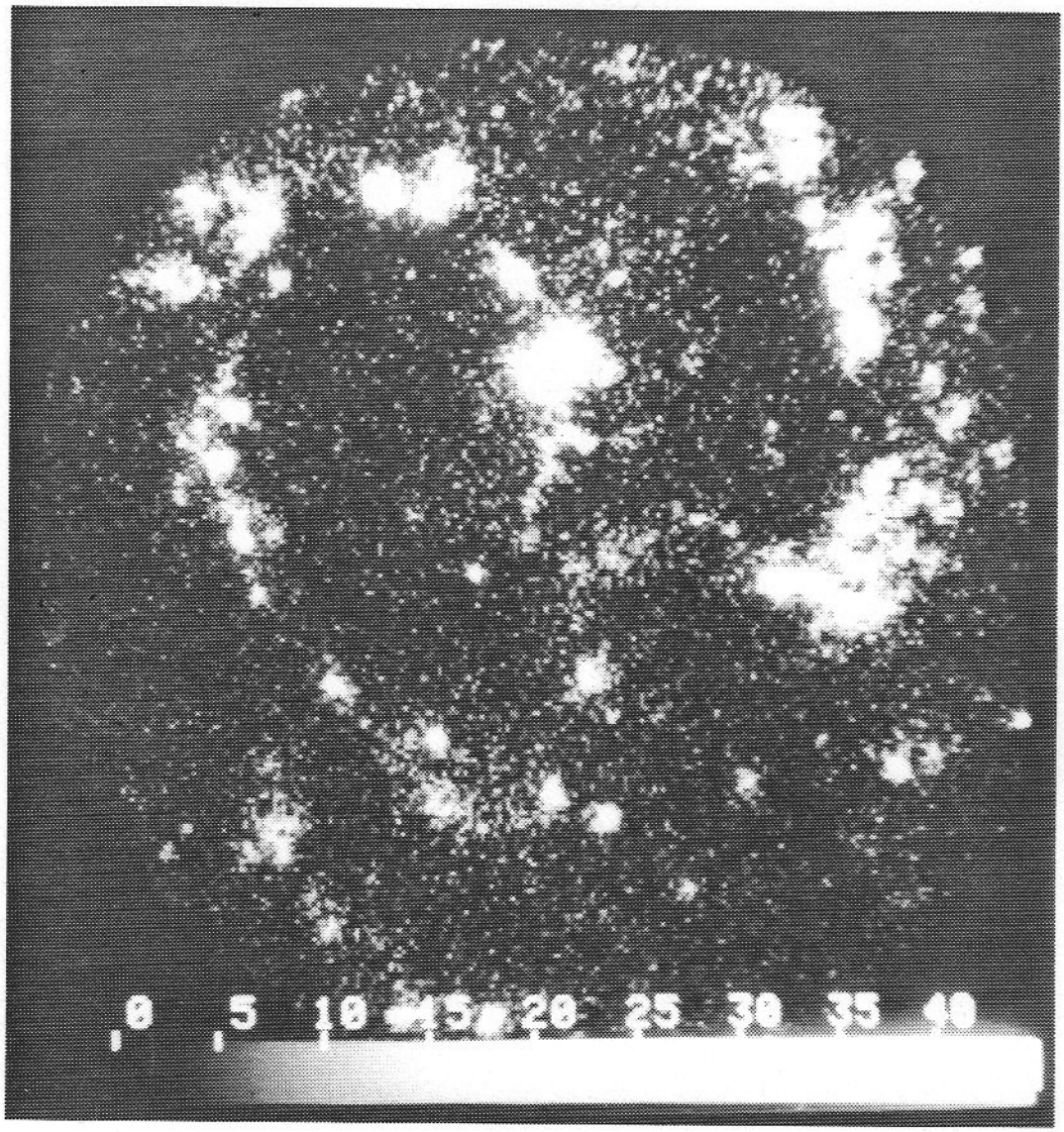

Fig. 2 Image of the total Hβ emission of M83, obtained from the data cube illustrated in Figure 1 by computing the zero moment along the velocity axis at each pixel position. The gray scale along the bottom of the figure indicates the total number of photons detected per 1.3 × 1.3 arcsecond pixel. From unpublished work by R.P.J. Tilanus.

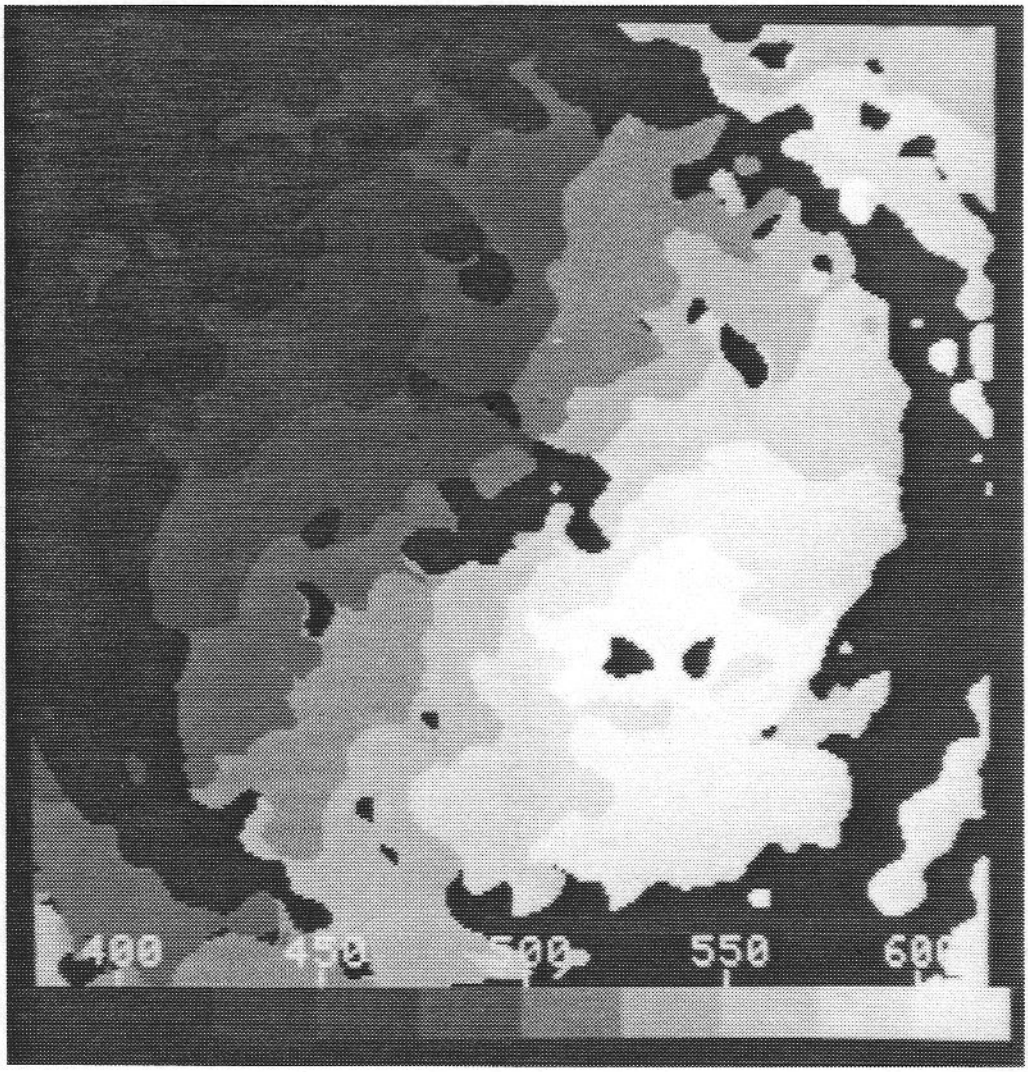

Fig. 3 Velocity field of the radio HI line emission from M83, as derived by computing the first moment along the velocity axis at each pixel position in a data cube obtained with the Very Large Array in the United States. The original data cube size was 512 × 512 pixels × 31 channels; it has been smoothed for this analysis to a resolution of 15 × 15 arcseconds × 20 km s^{-1}. A threshold technique has been used to exclude regions of insufficient signal from the velocity calculation. The field size of 12.8 arcminutes encompasses more of the galaxy than the Hβ image of Figure 2, but it still largely excludes a faint outer HI ring which is partly visible in the corners of the picture. The grey scale along the bottom is calibrated in km s^{-1}. From unpublished work by R.P.J. Tilanus.

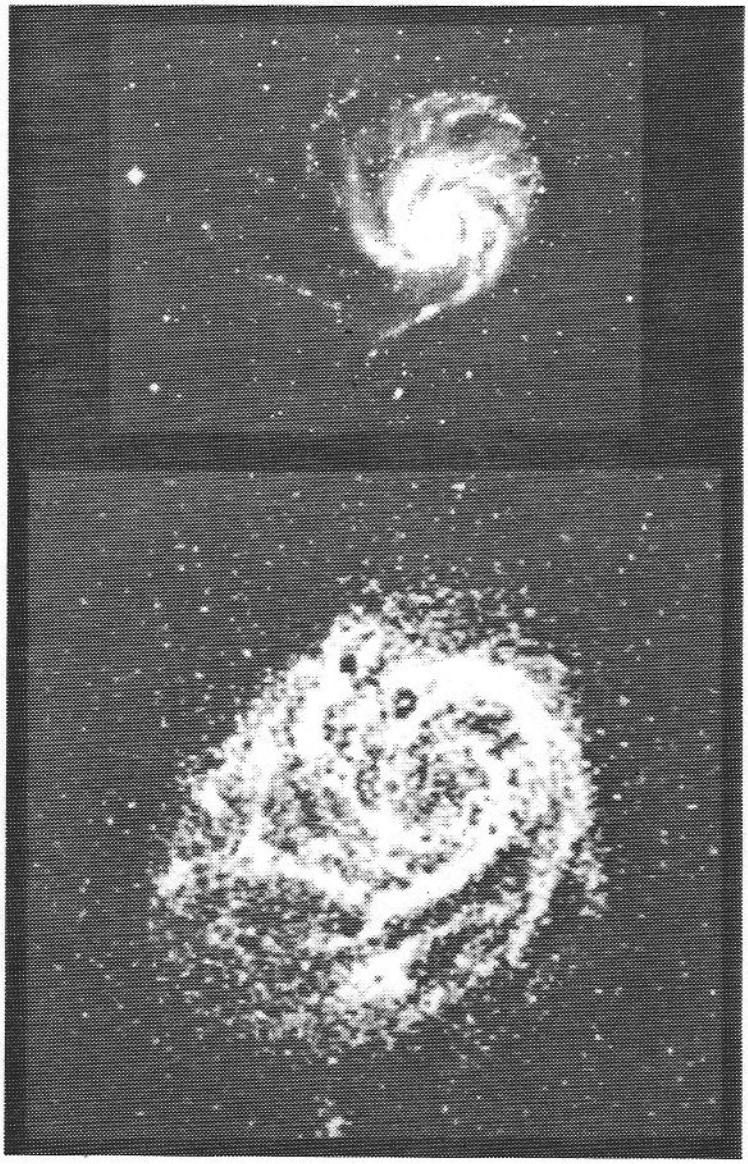

Fig. 4 Distribution of the radio total HI line emission in the northern spiral galaxy M101 = NGC 5457 (lower panel) compared to its optical image (upper panel) on the same scale. North is to the left. The original data cube was 512 × 512 pixels × 16 channels, observed with the Westerbork Synthesis Radio Telescope at a resolution of 24 × 30 arcseconds × 27 km s^{-1}. The HI picture shown here was obtained by computing the zero moment of the data cube along the velocity axis at each pixel position, using a thresholding algorithm in order to reject channels with insufficient signal. From Allen and Goss (1979).

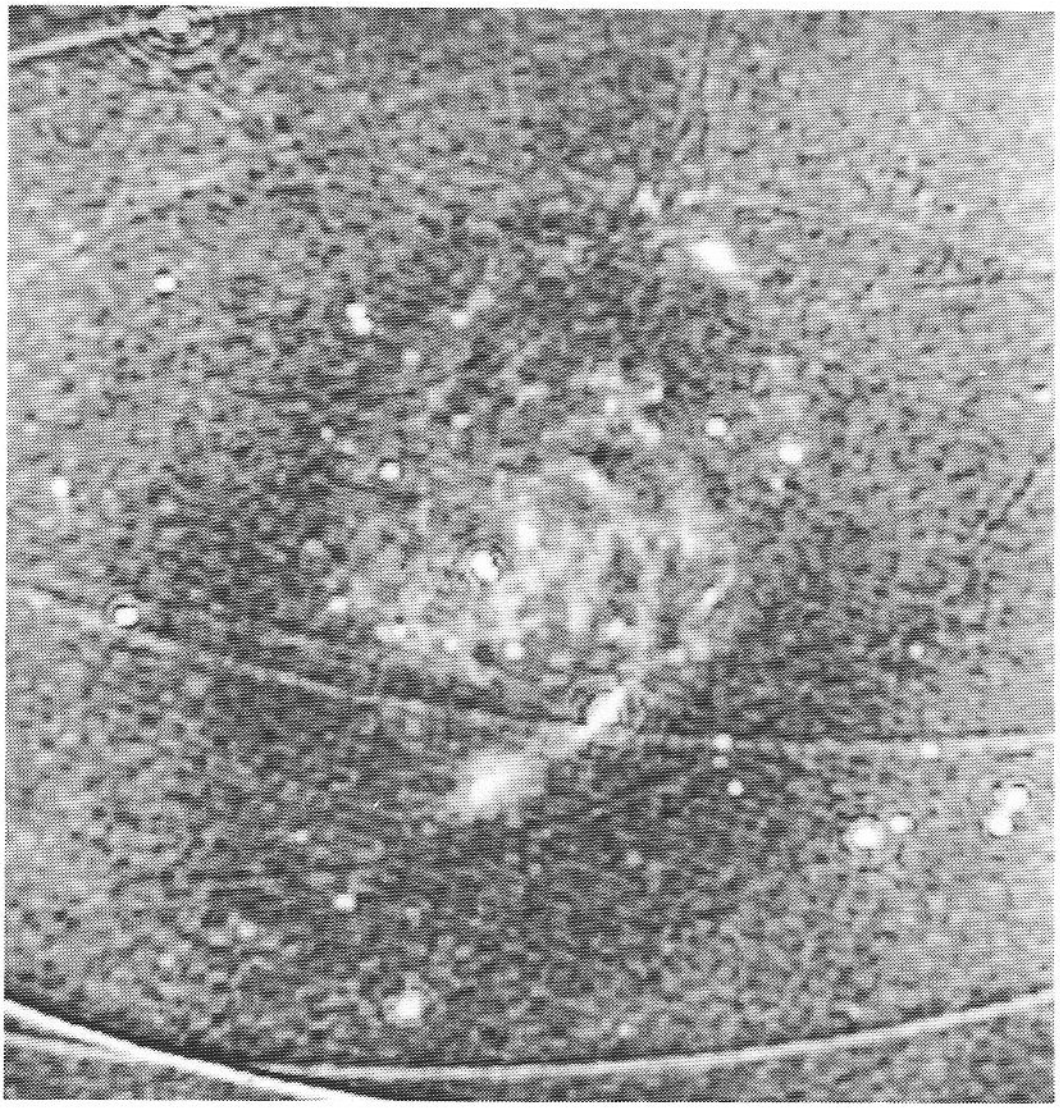

Fig. 5 Distribution of the radio continuum emission from M101, as observed with the Westerbork Synthesis Radio Telescope in the Netherlands at a wavelength near 21 cm. North is to the left. The resolution is 13 × 16 arcseconds. The arc-shaped features crossing the image and the small concentric rings surrounding discrete sources are instrumental effects which can be removed by further processing. This image can be compared with the optical picture in Figure 4, and with the old, much less sensitive result shown in Fig. 5 of Ekers et al. (1973). From unpublished work by Viallefond and Goss.

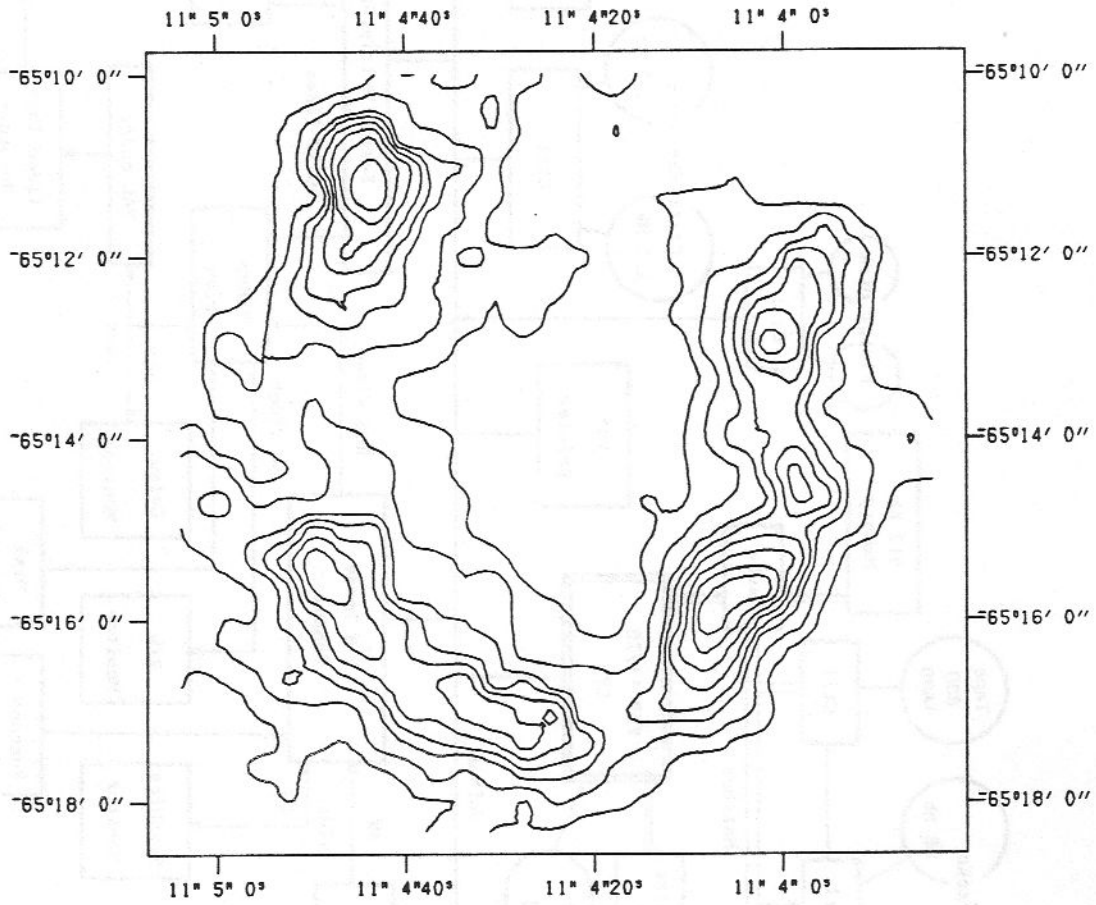

Fig. 6 Distribution of the infrared continuum emission at a wavelength of 50 microns from the galactic ring nebula RCW58, as observed with the Chopped Photometric Channel on board the Infrared Astronomical Satellite. The resolution is 1.5 × 1.5 arcminutes. The contours show emission by dust, which has a colour temperature of about 30 to 40K. There is a good correspondence with Hα features of the emission line images published in Chu (1982). The radius of the nebula is about 2.4 pc. From unpublished work by van der Hucht, Jurriens, Wesselius, and Williams.

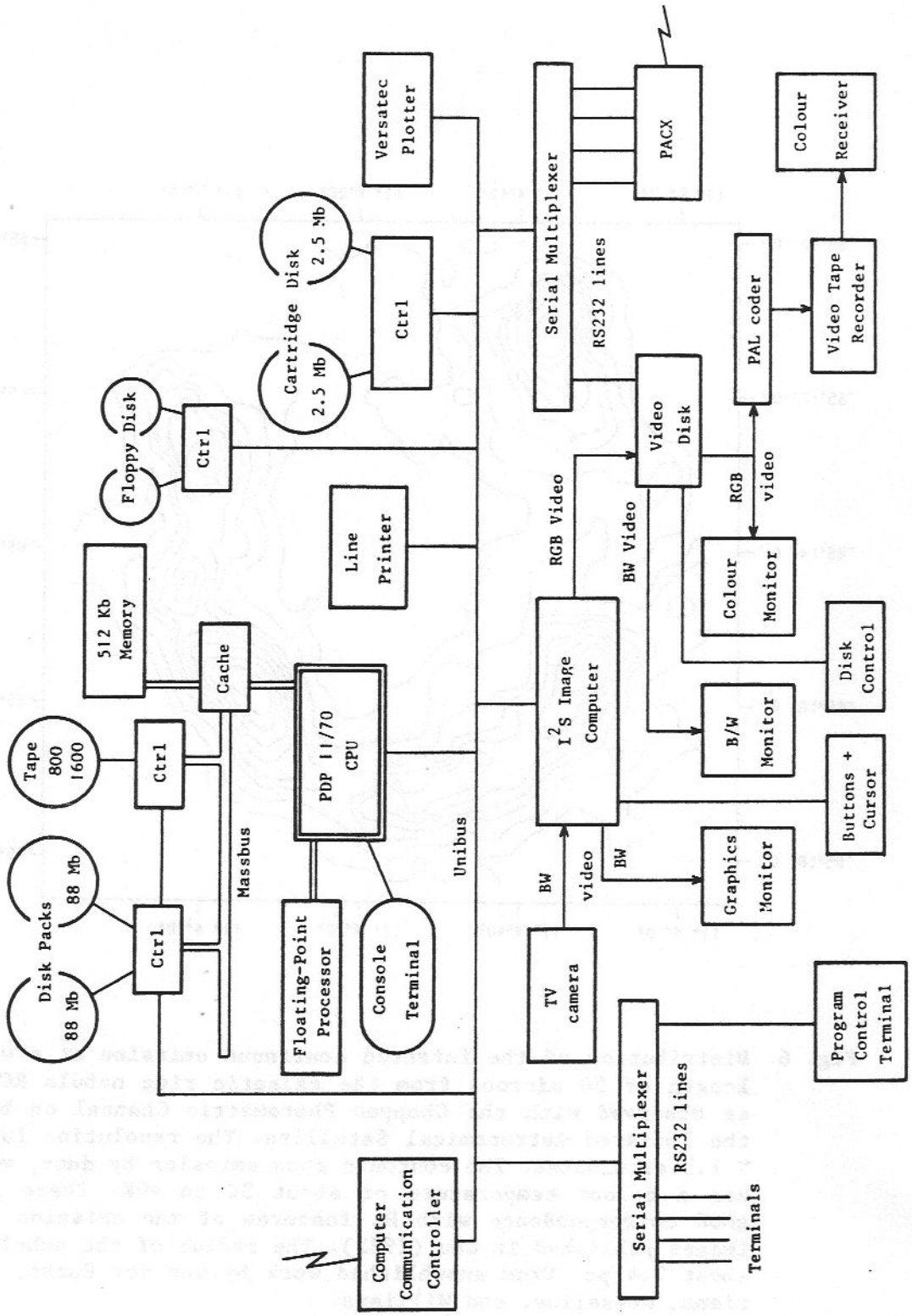

Fig. 7 GIPSY hardware (1983). Arrowheads denote analog signals.

colour, zoom, etc.) and cursor interactive functions, the device is capable of performing arithmetic operations on 512 × 512 8-bit and 16-bit signed and unsigned pixels at video rates (100 nanoseconds per pixel), and of feeding the results back into its own memories. The performance of the machine for such calculations has been evaluated previously (Allen, 1979), and an implementation of the CLEAN algorithm commonly used in radio astronomy has also been described (Allen, 1983). Storage of graphic images is provided in six memories of 512 × 512 1-bit pixels. Graphics can be overlaid on colour or grey pictures, or displayed on a separate screen.

Video Disk Recorder. This is an analog device[b] used for storage and cinematographic playback of processed images. It has a capacity of 100 colour or 300 grey pictures. Controls are provided for displaying sequences of pictures at various speeds up to the full rate of 25 frames per second. The video disk is used for:

- reconnaissance of large 3-dimensional data bases, by coding one of the principle axes as a time sequence;
- temporary storage of intermediate results for visual inspection during a lengthy data-processing session;
- blinking between two stored images in order to discover differences; and
- storage of processed images for later recording on film or on video cassette.

Electrostatic Plotter. This device[c] provides high-quality hard copy for graphics images such as contour plots and profiles, and working copies of half-tone images. Colour hard copy is obtained by photographing the display screen.

2. Software Architecture

The general structure of GIPSY software is analogous to (but simpler than) the combination of operating system, utility programs, and compilers found in any general-purpose computer. The major elements are:

- a master control program to handle user terminal commands and to initiate and monitor the application programs;
- a three-dimensional data base;
- a set of input-output interface subroutines to allow application programs to communicate with the user terminal and the data base;
- a structured high-level programming language;

---

b. Model RP3332B, Image Processing Systems, 70 Glenn Way, Belmont, CA 94002.
c. Versatec V-80.

- a standardized method for adding new program modules to the system library; and
- a structured documentation scheme.

Application programs. These are isolated from the user, data base, and master control program by a set of layered interfaces. This structure is intended to reduce the level of connectivity of the whole sytem, according to the precepts of the programming "principles" discussed by Allen and Ekers elsewhere in this volume. The general structure of a GIPSY application program is shown in Figure 8. The code written by an astronomer for his particular problem is symbolised by the inner dashed box (number 3) in this Figure; a structured Fortran-like language called Sheltran (Croes and Deckers, 1975) is used for most of our programming. Sheltran forces the programmer to write his code as a set of closed control structures, and provides additional facilities for improving the readability of the program listings. The application program communicates with the rest of the system through a small number of standard interface subroutines (dashed box numbered 2 in Figure 8). Finally, the program along with all of the interface and utility subroutines is built into an independent unit of executable code which requires only the master control program and the host computer operating system in order to run. In this way, faults in one application program are prevented from having a major impact on the system as a whole.

Master Control Program. This control program was introduced into GIPSY in 1979. The motivation was to improve and simplify the interface between the user and the applications program which he wishes to run. The design specifications have been discussed in more detail by Allen en Terlouw (1981). The main activity of this program turns out to be transferring messages between application programs and the user, between application programs and the operating system utilities, and among application programs themselves; hence its name "HERMES". Its services to the user are:

- to relieve users who are not computer programmers from the necessity of learning how to handle the computer manufacturer's operating system;
- to provide a simple overview on the terminal screen of the status of all application programs which are running;
- to standardize the appearance of all application program dialogue; and,
- to keep a "log file" of all activities and to allow the user to present any page of this file at random on his terminal.

The user services provided by HERMES will be further described later in the section on the user interface to GIPSY. We mention here several other features which were not originally basic design elements but which have turned out to be significant:

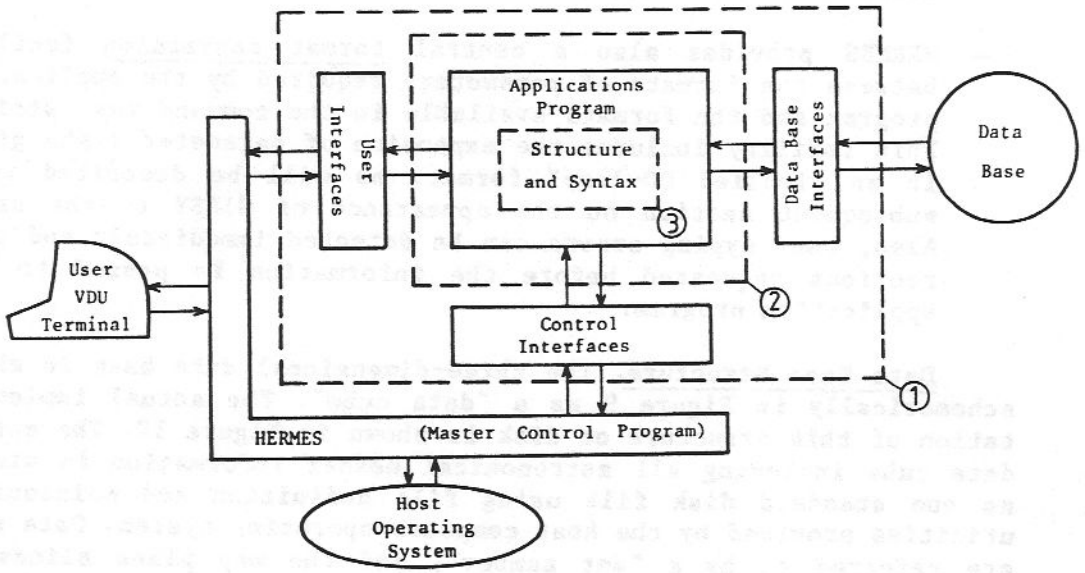

Fig. 8 GIPSY applications program structure.

- HERMES is in a position to keep and update command strings for each application program which it has run. These text strings contain the most recent values for parameters in the form of all KEYWORD = VALUE pairs typed by the user for a particular application program. A macro facility can now be provided by giving this string an abbreviated name. Application programs can then be easily re-run with parameters from the last execution plus any desired current modifications.

- Viewing the message traffic in the other direction, HERMES is also in a position to analyse requests coming from an application program and decide whether to present the query to the user on his terminal screen or to provide a reply from some other source. This opens the way for a rather elaborate scheme of defaults for parameters, which may be conditioned on the data and on parameters obtained previously. These defaults are controlled from within the application program, as will be described in the section on the user interface subroutine to follow.

- Since HERMES interprets user terminal input on a character-by-character basis, it can respond to special single-keystroke commands (which are not echoed on the screen). For example, the "TAB" key causes the display of the previous page in the log file on the terminal screen, while "control F" starts a program to produce a grey representation of the display screen

on the electrostatic plotter. A complete list of these commands is given in Appendix A.

- HERMES provides also a central format conversion facility between the formats of parameters required by the application program and the formats available in the command text string. This facility includes the expansion of parameter lists given in an "implied DO-loop" format, as will be described in a subsequent section on the appearance of GIPSY to the user. Also, user typing errors can be detected immediately and corrections requested before the information is passed to the application program.

Data Base Structure. The three-dimensional data base is shown schematically in Figure 9 as a "data cube". The actual implementation of this structure on disk is shown in Figure 10. The entire data cube including all astronomical header information is stored as one standard disk file using file definition and maintenance utilities provided by the host computer operating system. Data sets are referred to by a "set number", and the map plane slices (z axis) of the data cube by a "subset number". The implementation is a compromise between conceptual generality, speed of access, and simplicity.

Interface Subroutines. GIPSY interface subroutines are divided functionally into three categories: user, data base, and control. We discuss here as an example one major subroutine from each category; a complete list is given in Appendix D.

USRINP: This subroutine is used for obtaining parameters from the user. It provides the programmer with the following functions:

- interrogates the user terminal for a (list of) parameter(s) if that parameter (as designated by its keyword) is not otherwise available;
- provides the parameter in a format chosen by the programmer, independent of what the user actually typed;
- handles all possibilities for default values of parameters;
- gives immediate syntax error messages to the user, so that the returned parameters are at least correct in that respect;
- saves "keyword = parameter" pairs for subsequent use in the application program; and,
- adds text strings recording these transactions in the log file.

The subroutine is called by the application program as:
CALL USRINP (ARRAY, NELS, TYPE, LENGTH, DEFAULT, KEYWORD, MESSAGE)
The arguments are described in Appendix E, where the documentation for this subroutine is presented as an example. KEYWORD is a text string used as a prompt, as a label for the parameter string given by the user, and as a label to store parameters in a "macro" to be

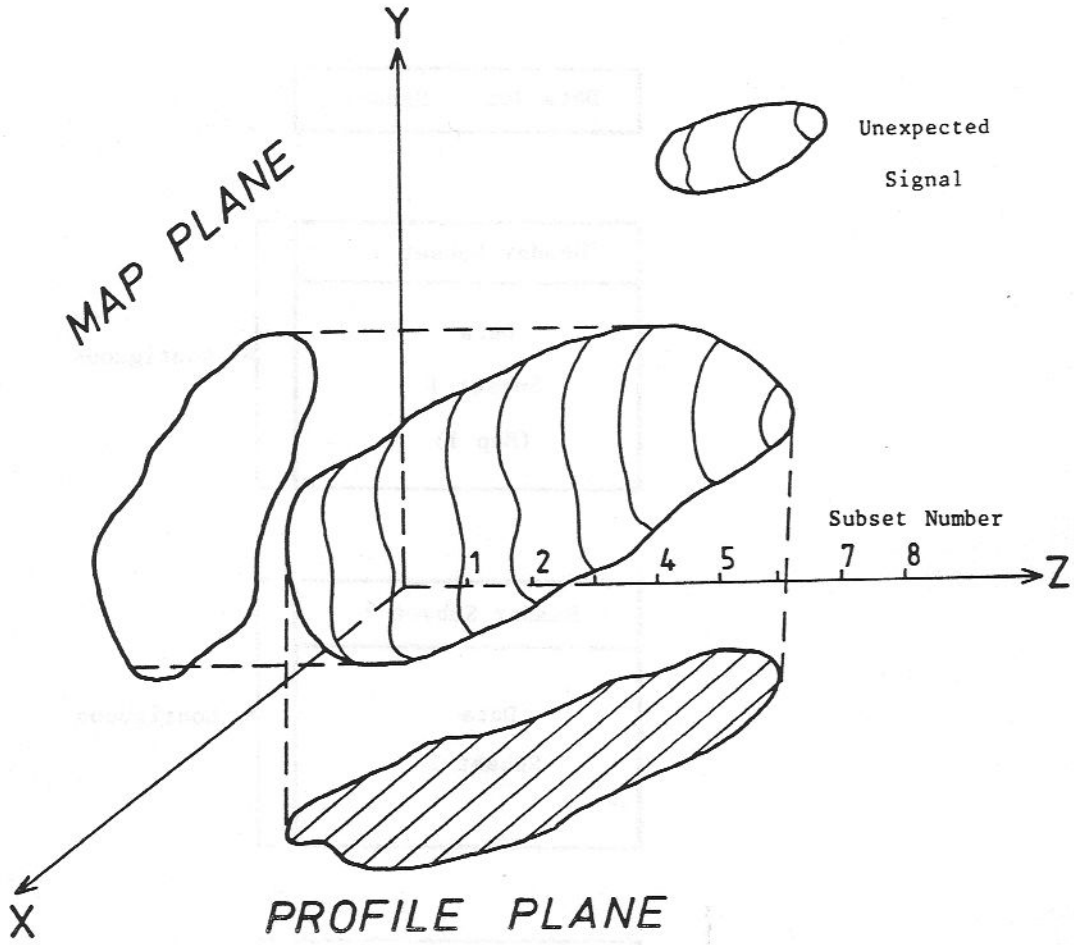

Fig. 9 The three-dimensional data structure or "data cube" of intensity or number of photons.

used as defaults in a future execution of the program. DEFAULT is an integer defining what kind of defaults the applications programmer will accept:

- DEFAULT = 0 means no default is possible. The user is prompted and must reply with a parameter. An example is when requesting the set and subset numbers of an image.

- DEFAULT = 1; the user is prompted, and if he replies with a carriage return the current value of the parameter (preset by the programmer before calling USRINP) will be returned as a default. An advantage of this construct is that defaults can be computed from within a running program, based on previous-

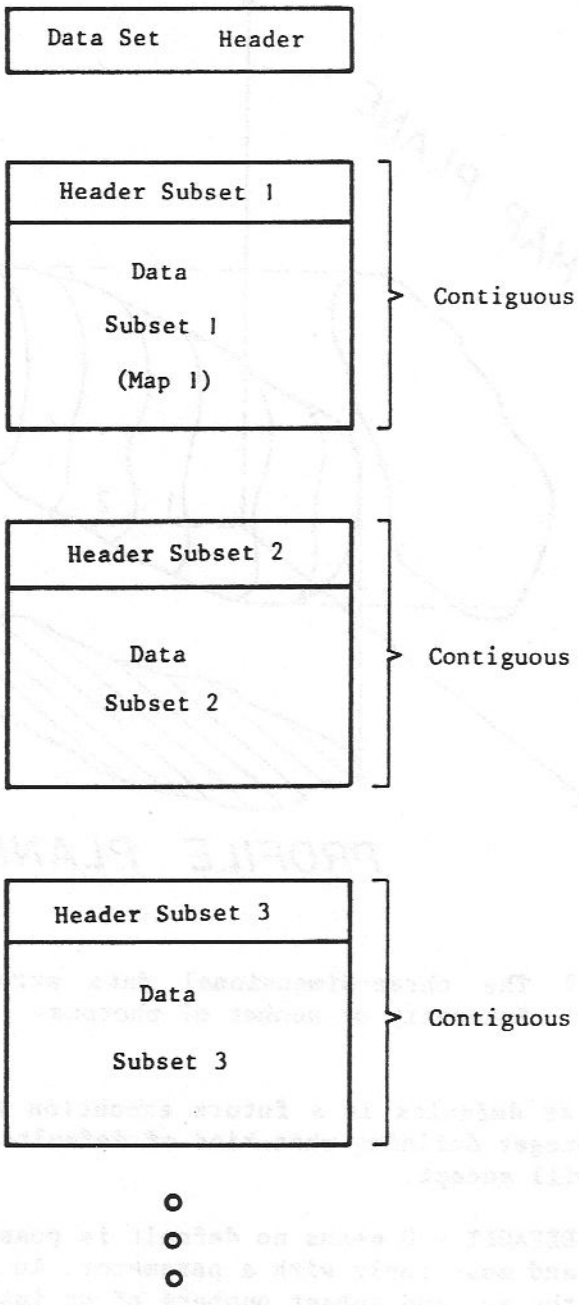

Fig. 10 Implementation of the data base as one file.

ly-entered parameters and on the data itself. An example is to default to the full size of the image (determined by the program from reading the image header) for some numerical operation which may in general be done on a restricted area.

- DEFAULT = 2 is intended for all parameters which the average user does not normally need to know about or to change. If it really is necessary to change them, this can be done by prespecifying the keyword and parameter list any time before the program arrives at the particular call to USRINP; otherwise the application-program-selected defaults are automatically taken. Keywords with this level of default are often referred to as "hidden keywords", since under normal circumstances no prompts appear on the user's terminal. They are also very useful for testing programs.

Note that the user may give a parameter string at any time. The master control program saves the information as an ASCII string until the applications program requests it.

The default facilities described above, coupled with the capacity of the master control program to save command strings, provide a high degree of flexibility for both novice and experienced users of GIPSY. Novice users usually wish to use only the standard features of an application program; they appreciate being extensively prompted for relevant parameters, and are happy to have the program choose sensible defaults by itself. Also, they do not want to be asked for parameters which are not necessary. More experienced users quickly find such prompting to be a nuisance, and prefer to prespecify parameters or use a private set of defaults defined in a text string when starting a program. In this way they can also use special features of the program accessible via hidden keywords.

Flexibility in the interpretation of defaults is also available to the GIPSY programmer. For instance the "carriage return" default = 1 can be taken to mean different things in different parts of a program even if the keyword is the same. As an example, the program DISK begins by asking the user to provide the keyword SET=; carriage return is interpreted to mean that the user wants an abbreviated list of all data sets available to him. After providing this list, the program then prompts for SET= again, but now a carriage return stops the program. In both cases, if the user replies with a specific set number the program gives him a description of all subsets contained in that data set.

READXY: This subroutine is used to read a piece of the data cube from disk storage in the map plane (see Figure 9). The data set is specified by a SET number, and the location on the z-axis by the SUBSET number. The call is:

CALL READXY (SET, SUBSET, XBEG, YBEG, ARRAY, NX, NY, IER)

The frame to be extracted from the given subset is defined by its beginning coordinates XBEG, YBEG, and size NX, NY. The data is returned in ARRAY and an error code in IER. NY may be 1 if only one line of data is required. The coordinate system used for subsets is the normal, right-handed system which one learns in primary school, with the point (0,0) at the center of the image; XBEG, YBEG therefore refer to the lower left corner of the frame to be extracted. GIPSY application programs normally do not concern themselves with opening and closing files of image data. The interface and control routines handle this automatically; if not already open, a data file is opened at the first access request and closed when the program terminates. Finally, we note that pixel values are stored on disk and manipulated in programs as floating-point numbers. This entirely avoids requiring programmers to pay attention to integer overflow problems (which were a headache in our first-generation system), or to write both fixed- and floating-point versions of their code.

DEPUTY: This subroutine allows one application program to run a second application program, using the parameter list of the first program. The first program may add to its parameter list any keywords which it wishes to pass using the subroutine WKEY before calling DEPUTY. It may be convenient (or even necessary) to rename keywords using the subroutine SUBST before calling DEPUTY. For instance the output data set identified by the keyword SETOUT= can, by this means, be associated with the input SET= of a subsequent program started with DEPUTY. The call is:

CALL DEPUTY (TASK),

where the one input argument is a text string giving the name of the application program to be started. Execution of the first program is normally suspended until the second program has finished. The second program may transmit parameters back to the first by using WKEY, provided the keywords are then requested (by calls to USRINP) after the first program has resumed.

Notes on keywords and control subroutines. The parameter string for a given application program is private to that program; GIPSY keywords are local. This is done in order to minimize the connectivity of the system. Programs which are started by other programs do share keywords; however, the system treats such "spawned" programs as a single entity with the name taken from the first program. The user is therefore confronted only with that name, and is unaware that the work may be carried out by several different application programs in succession. A program failure at the end of the chain causes HERMES to abort all other programs in that chain.

Updating the software library. Programs which are ready to be made available for general use are inserted into the system library by means of a standard procedure. This procedure, which may be operated by the programmer himself, is automatically started by

logging on the host computer operating system with a special password. The procedure is set up in an interactive program which requests the names of the input files, processes these through compilers, builds the task (executable program) if necessary, and writes the results to the program library on disk and to backup media. It also makes entries in a history file where all permutations on the system library are recorded. After this procedure has been run successfully, the programmer deletes all files pertinent to that program which he may have in his own disk space. This saves on space, but also guarantees that when someone else may wish to modify that program he may be confident that the system library contains the most recent, operable version of the source code. The update procedure also provides for deleting programs and documents which are no longer required.

Documentation scheme. This scheme is divided into three levels, designated 1, 2, and 3:

- Level 1 documents provide general information and descriptions of how to operate the various application programs. Users need only be familiar with documentation at level 1;
- Level 2 documents provide descriptions of all interface subroutines to the user terminal, data base, and for control. At this level we also find various utility subroutines and slave tasks, and descriptions of how to write level 1 programs;
- Level 3 documents describe software which is specific to the particular configuration of computer hardware and host operating system currently in use. Programs at this level are usually not explicitly called by an astronomer who is writing an application program; they are, of course, called by the level 2 subroutines which he uses.

Documentation examples are given in Appendices C and E.

IV. HUMAN INTERFACES TO GIPSY

As discussed by Allen elsewhere in this volume, a data processing system presents a different appearance to different people depending on whether the system is viewed by a user, a programmer, or a system manager.

1. The User's View

An astronomer who plans to use GIPSY for his data analysis may begin by reading a set of documents collected in the "New User's Documentation Package". Time on the system for production processing is normally allocated to users in blocks of several hours long. Each major user (or guest) of GIPSY receives a private disk pack which he may use for his own data processing; these packs can hold up to about 80 maps of 512 × 512 pixels. After installing his disk pack on the drive, he sits at the control console facing a standard

video display terminal and keyboard (VDU) and flanked by a number of television screens. The user station is shown in Figure 11. Logging on the host computer with a special password initiates an automatic start-up sequence which, besides performing a number of housekeeping functions, presents the user with the latest page of the Daily News. This is a computerized text file containing commentary on software bugs and on the latest modifications to the system. The master control program HERMES then takes over, re-organizing the VDU screen from the conventional line-by-line format to the format sketched in Figure 12. The subdivisions are:

- the "User Command Area", consisting of two lines at the bottom of the screen. All printable characters typed by the user are echoed here;
- the "Task Status Area" consisting of 4 lines showing the status of currently active tasks (i.e. application programs);
- the "Common Output Area" consisting of the top 18 lines of the screen. This section is used to display pages of the log file, showing information sent from application programs.

The Common Output Area (COA). The COA shows one page of 18 lines from the log file. These pages are numbered, and on the right edge of the screen in the Task Status Area are shown the page number currently being displayed on the COA, followed by the current page number on which HERMES is writing messages. There are two modes of displaying the log file information: "page mode", and "non-page mode". In non-page mode (the default on startup) the current- and display-page are always the same, i.e. if a new page is initiated, the old one disappears from the screen. In page mode the information on the screen will stay there until the user instructs the system to change the page. There are two classes of commands to manipulate the COA: control characters; and commands terminated with <CR> (carriage return). The control characters always have an immediate effect, regardless of the state of any line currently being typed, and are not echoed in the User Command Area. These page control characters and commands refer only to the display, and do not inhibit tasks from continuing to write in the log file in the normal way. Commands of both types are listed in Appendix A.

The Task Status Area (TSA). This is the middle section of the terminal screen consisting of four lines which may contain status information of "active servant tasks"; they are the application programs requested by the user, or utility programs initiated by HERMES. To the extreme right of the TSA a time-of-day clock and the current- and display-page numbers are given. Some examples of the status information are (asterisks shown blink on the screen):

- When a task has been activated, but is not running yet;
  taskname WAITING TO BE RUN
- When a task is running;
  taskname RUNNING

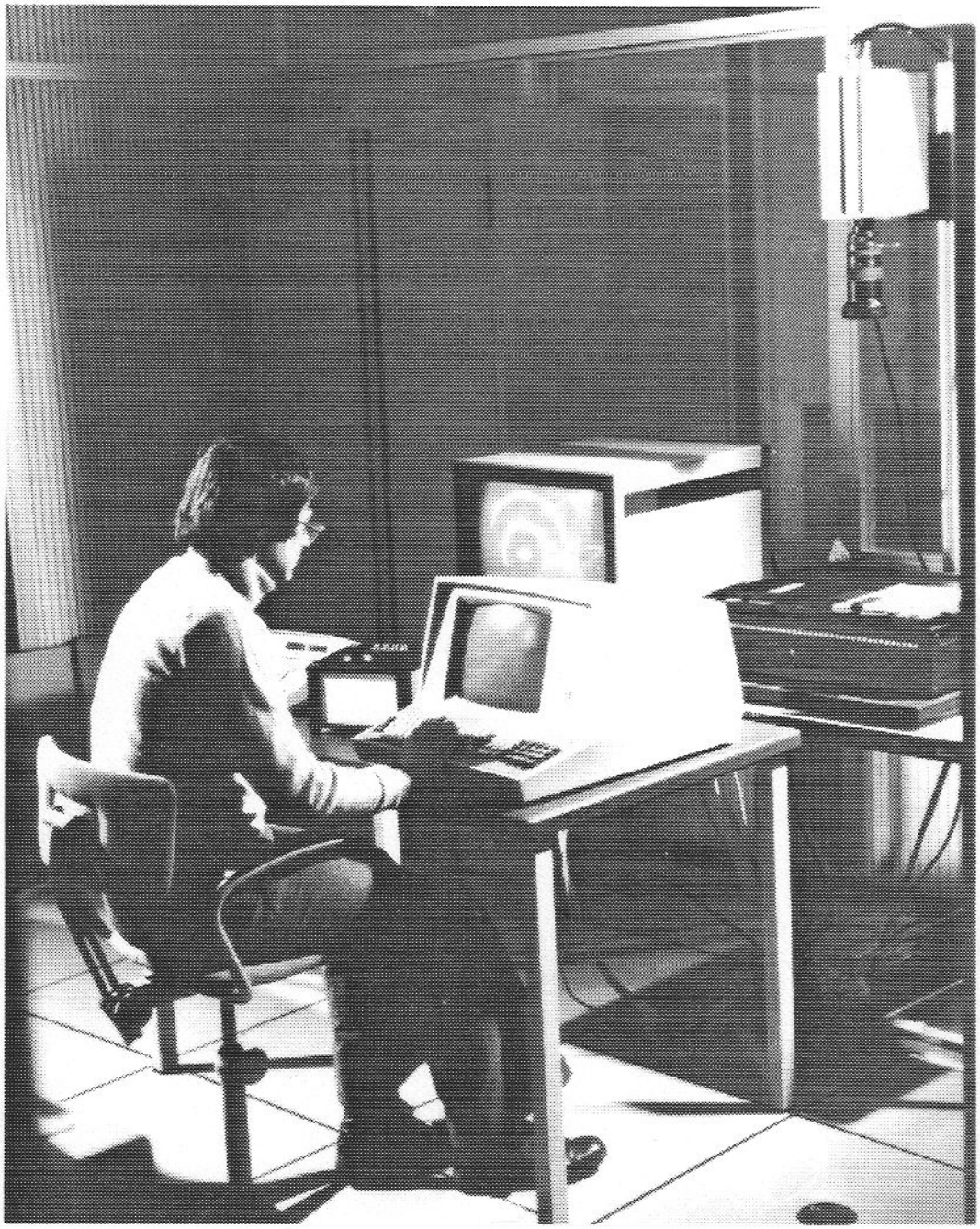

Fig. 11 GIPSY user station.

```
<USER  >TRANS ,SETOUT=
Output set and subset(s): 0,1
<USER  >TRANS,SET=6 1,SETOUT=
Input  set and subset(s): 6 1
Output set and subset(s): 0,1
<USER  >TRANS /WAIT
<USER  >TRANS/GO
<USER  >ROTATE
<USER  >ROTATE,SET=6 1
<USER  >ROTATE,SETOUT=
<USER  >ROTATE,SETOUT=0 1
<USER  >ROTATE,POS=
Set    6,  1 from (-127-127) to ( 128 128)
<USER  >ROTATE,AREA=
 COTRAS:instrument unknown. WSRT assumed.   **WARNING **
              L=-127 TO 128   M=-127 TO 128
<USER  >ROTATE,ANGLE=10

- ROTATE Set   0,  1;   8 percent of map ready                    12:10
- PRINT     * default is 30 * 50 if only the center is given      549  551
- CPLOT     * Type set and subsets (0,1)
- LOOK    USER ABORT
PRINT ,AREA=_____________________________________________________________
 _____________________________________________________________________
```

Fig. 12 User VDU terminal screen format.

or
taskname status message (supplied by servant task)

- When a task is pausing;
  taskname *PAUSING (optional message)
- When a task requires input;
  taskname *input request message from that task
- When a task encountered a fatal error;
  taskname error message -FATAL
- When a task has crashed (i.e. stopped without notifying HERMES);
  taskname *CRASHED
- When a task has finished processing normally;
  taskname +++FINISHED+++
- When a task has been aborted by the user;
  taskname USER ABORT

The last four messages will disappear as soon as the user initiates another servant task or when the user types ↑R. The command ↑R can also be used to recover the TSA if it is accidentally destroyed.

The User Command Area (UCA). The User Command Area (UCA) consists of two lines located at the bottom of the display screen. It is logically one long line, with 148 typing positions available for user input. Some examples of the commands that can be typed will be described later. HERMES itself can also "type" in this area. It does this for instance in the following cases:

- when it is likely that a "taskname,keyword=" combination will have to be typed, HERMES types it for the user;
- when the user types ↑Y or ↑T, a cursor processor is activated. When the cursor position has been read out, the coordinate values are typed in the UCA and can therefore appear as keyword parameters for input to another (active or still to be activated) task.

Text present in the UCA which is either erroneous or otherwise unwanted is simply deleted with the DELETE key (or ↑U to erase the whole line). For instance in this way a task waiting on input via a "taskname,keyword=" request in the UCA need not be satisfied before initiating some other task. ↑U erases the text, and a new command can be given by the user. If the UCA is empty, striking the space bar causes HERMES to cycle through presenting the currently unsatisfied keywords of active tasks.

At the end of the UCA, HERMES displays error messages which are related to the line typed by the user (e.g. BAD SYNTAX, ALREADY ACTIVE, NOT PRESENT, etc.). When a line is accepted by HERMES, it will disappear from the UCA; if it is rejected (as can be seen from the error message) it stays on the display, but will disappear as soon as the user starts typing a new line. Rejected lines are not displayed in the COA and do not appear in the log file.

Finding the documentation. Copies of all user documentation are kept in loose-leaf binders at the control console. This documentation includes general information as well as descriptions of all Level 1 application programs. Copies of these documents can be produced on the VDU screen and printed in the log file by entering the command:

HELP TASK= program name.

A one-line description of the application programs currently available in GIPSY is given in Appendix B. Several frequently-used tasks can be initiated using the (programmable) function keys located across the top of the VDU keyboard.

Giving Parameters to Tasks. If the user does not pre-specify any parameters when initiating a task, the task will prompt him for the necessary information as it proceeds or else take default values as indicated in its documentation. As the user becomes more experienced the screen dialogue and delays involved in answering these prompts can be circumvented by pre-specifying parameters, for example as follows:

PRINT SET= 12,4 AREA= FC DG 20 20

This command will initiate a task to print the pixel values from an area of 20 × 20 grid points centered on the field center of the map in set 12, subset 4. The KEYWORD-PARAMETER pairs may be given in any order, separated by commas or blanks. HERMES will save the keywords, giving the associated parameters to the application task when it requests them. Several frequently-used keywords such as SET= and AREA= are available as single-key commands by means of the row of function keys across the top of the terminal keyboard. A task which has been previously run can be repeated with the same parameters by typing @TASKNAME. Any selected parameters the user wishes to change can simply be appended; for instance if the user wishes to repeat the execution of PRINT described above but on a different map he need only type

@PRINT SET= 6,3

which will print the pixel values from the map in set 6, subset 3, over the area given in the immediately previous initiation of the task PRINT. Note that the KEYWORD=PARAMETER pairs for a given task apply only to that task and not to any another task, even though the keywords for different tasks often have the same names.

Formats for Input Parameters. The format for input values is completely free; HERMES takes care of converting the numbers into the formats actually required by the application task. Any of the legal Fortran formats are accepted; otherwise list elements are assumed to be character strings. If a character string has been requested by a task, no conversion is made. For real and integer values the user may give a repetitive list in the form s:e:i where s = starting value, e = ending value, and i = increment. For instance the keyword for contour levels

CNT= -3:10:2

means the list of contour levels -3, -1, +1, +3, +5, +7, +9.

Coordinates which have been requested as parameters can be given to any task in a variety of forms, as specified by a prefix followed by a blank. For instance

POS= * 13 27 6.4 23 34 10

designates a position at right ascension (RA) = $13^h27^m6^s.4$, declination (Dec) = +23° 34' 10". The prefixes currently available are:

- G grid units in RA and Dec with respect to the map center, e.g. G 10 -12. This is also the default if no prefix is given;
- * celestial coordinates RA and Dec in hours, mins, secs and degrees, mins, secs;
- D RA and Dec in degrees;
- O RA and Dec in degrees offset from the map center;
- RP polar coordinates from map center in degrees;
- FC celestial coordinates of the map center; and
- LB galactic coordinates in degrees.

A rectangular framed area of a map may be specified either as the lower left and upper right corner or as a center and size; HERMES knows which of the two you mean. For instance:

```
AREA= -10 -12 +4 +6.
or
AREA= -3 -3 DG 15 19
```

both refer to the frame with lower left corner X,Y = -10, -12 and upper right corner +4, +6. Two further prefixes are available for areas:

DD size in degrees of RA and Dec; and
DG size in grid units.

For instance the command

```
AREA= FC DG 11 11
```

is equivalent to

```
AREA= -5, -5, 5, 5
```

and refers to an area of 11 × 11 grid points located at the map center.

Using the Cursor. A very convenient way of obtaining position and amplitude information from an image is to use the cursor, moving it by means of the trackball and selecting the desired parameters with the function keys on the cursor box. The cursor task can be started with ↑Y or ↑T. The first cursor is a small cross which comes equipped with the grid coordinate values of the pixel in the image under the cursor; the second is a "full screen" cursor to be used with vector plots such as contour diagrams. Information about the coordinates and amplitude of the pixel under the cursor is continually updated in the task status area. Depressing function keys on the cursor box causes parameters to be "typed" in the user command area, as follows:

- button A gives the current grid coordinates of the cursor in the order X, Y;
- button B gives the coordinates in right ascension and declination;
- button C gives the pixel amplitude; and
- buttons A and B together terminate the cursor task.

Note that the parameters entered into the UCA in this way are the same as if the user had typed them himself; characters may be erased (with the DELETE key) and new characters inserted until the desired values are obtained.

Finishing the session. Since the results of his data analysis are kept on his own private disk, the user may easily interrupt his computer work and return to it in the future. The session is closed by initiating the task BYE, which also will take care of saving, or printing and deleting, the log file accumulated during the analysis session. If kept, the log file remains on the user's private disk; at the following startup, the system will continue at the same place it left off.

Feedback from the users. It is virtually impossible to test each application task for reliable operation under all combinations of input parameters and data. Failure of specific tasks, or suggestions for improvements of their operation or documentation must be brought to the attention of the person currently responsible for that task (see the documentation of the task) by using a Software Failure Report. A stack of blank forms is available at the GIPSY user console. The original, along with all supporting documentation (e.g. relevant pages from the log file), is to be given directly to the person concerned; the carbon copy is to be added to the stack kept in the accompanying folder at the user console in order to indicate to the System Manager and to subsequent users that the problem has been reported. All active users of GIPSY are strongly encouraged to attend the weekly Software Lunch Meetings, where suggestions for improvements and current problems are discussed.

2. The Application Programmer's View

Although there are currently about 180 application functions available in GIPSY (concentrated in 155 programs, cf. Appendix B), the methods of data analysis become more complex with the experience gained and there is continual pressure to modify and improve existing programs. In addition, GIPSY is now being used for the analysis of optical and infrared astronomical data as well as the radio astronomical observations for which the first set of application programs were designed, and new software is required for this. Many users therefore become programmers; in this section we describe GIPSY from that point of view. The first step for a new programmer is to read over the contents of "The New Programmer's Documentation Package".

Presenting the Problem. It's important that a proposal to write a new program be discussed with other astronomers using and programming GIPSY. The reasons for this are:

- the required function may already be available, perhaps as a combination of other functions;
- someone else is already working on the problem; or
- others need such a program too, and would like to offer suggestions about how to write it.

The regular weekly Software Lunch Meeting is the occasion to present a proposal for a new program. Such a proposal is best made in the form of a draft of the user documentation for the program, describing its function and all the keywords. This gives others a chance to propose improvements before such changes would require rewriting the code.

Software Conventions. Programs written for GIPSY are expected to conform to a number of conventions and standard practices which have been adopted in order to improve the maintainability and

durability of the software:

- use a structured programming language. For those who are used to programming in Fortran, the Sheltran language is an excellent alternative which is quickly learned and strongly supported in GIPSY; and
- become familiar with the function and features of the standard level 2 subroutines which are used to interface GIPSY application programs to the data base, to the user, and to HERMES.

The level 2 interface subroutines are listed in Appendix D.

In addition to these general practices, there are a number of specific syntactical conventions to be observed. Some examples of these are:

- make frequent use of subroutines, or of Sheltran procedures if subroutines are not convenient. They improve the modularity of the program and make it easier to read;
- avoid the use of COMMON to transfer information to private subroutines;
- subroutines which are private to the program should be kept in the same source code file as the main program itself;
- use PARAMETER statements to define array dimensions, buffer sizes, FOR loop limits, etc. This facilitates modifications in the future;
- avoid EQUIVALENCES;
- declare the TYPE of all variable and arrays explicitly; and
- do not use multiple ENTRY or RETURN points in subroutines.

An example of a GIPSY application program is given in Appendix E.

Documentation Conventions. As we have suggested earlier, the user documentation for a new application program should be the first thing to be written, before the program is coded. The standard format for the user documentation is given in Appendices C and E. Descriptions of how the program works belong in the code of the program itself; see Appendix F for an example. As a good rule of thumb, there should be as many lines of description in the program as there are lines of executable code.

Coding and Testing. An applications programmer receives a private password for the computer and a modest allocation of disk space on which to develop the software. Three source files have to be constructed:

- the application program SHELTRAN code;
- a command file with the instructions for building the task;
- the final user documentation.

When the program has been compiled and the first task successfully built, a test is needed. This takes priority over production data analysis during working hours. Getting HERMES to run a test program is easy; the program name is merely preceded by the programmers own identification under which the test version of the

application task resides. If full error message reporting is desired in order to catch difficult bugs, this can be switched on at start-up by choosing the appropriate answer to the start-up question.

The Referee System. Software written for GIPSY is subject to peer review; a referee can be selected by the program author. Referees examine the source code and the accompanying user documentation for comprehensibility; they are not required to check the program operation at the user console.

Updating the Software Library. This topic was described previously in section III.2 on the software architecture. After the procedure has run successfully, the programmer is given the opportunity of adding some prose to the DAILY NEWS in order to inform others of the changes and additions. Since all the source versions are now safe on the backup disks and the task itself is in the GIPSY library, the programmer can delete all files pertaining to that program under his own identification. This will be required anyway, since the number of blocks available to him on the system disk is too small for more than a few software projects.

3. The System Manager's View

The main job of the System Manager is to promote communication amongst the users and programmers. There are a number of mechanisms in GIPSY for accomplishing this:

- the list of current software projects;
- the Daily News;
- software failure reports;
- the weekly software discussions; and
- the referee system for new software.

Except for the first item, these topics have been mentioned earlier. The list of current software projects is an important working document for the system manager. It is a computer text file which he maintains, with a brief entry describing the nature, current status, and name of the persons responsible for new programs under development and for improvements to existing software.

In the course of the development of GIPSY over the years we have found it useful for the system manager to have the close assistance of two colleagues with rather different specializations:

- a professional programmer familiar with the operating system of the host computer but not concerned with analysing astronomical data; and
- a senior user intimately involved with astronomical applications but not concerned with writing software. This person acts as the "friend of the users".

V. AN EVALUATION

As we have stated in the introduction, we view the present version of GIPSY as the product of more than 13 years of continuous development. We are now on the verge of moving to our fourth host computer and operating system, and it is an appropriate moment to sum up the current status.

1. System Size and Cost

An impression of the size of the system software can be obtained from Tables 1 and 2, which provide a breakdown according to level and type of the programs and documentation. We continue to invest manpower at the rate of 3 to 4 man-years per year, a typical value since the start of the project. The average maintenance and development capacity is about 35,000 lines of source code plus documentation per man.

The hardware and software costs are summarized in Table 3, integrated over the expected 8-year lifetime of one generation of computer hardware. In this time span, the hardware and software costs are about equal.

2. Good Features of GIPSY

From the experience of users and programmers we note the following positive points about GIPSY:

- there is a large number and great variety of application programs, and the system has the flexibility to allow them to be combined in novel ways;

TABLE 1
Breakdown of GIPSY software on the host computer system disk as of 10 October 1984 by level, including binary tasks, (level 1 only), source code, object libraries, and documentation:

Level	Description	Number of Programs	Total Size (Megabytes)
1	Application Tasks	155	13.2
2	Interface and Utility Subroutines	116	0.8
3	Hardware and Operating System Subroutines	96	0.3

TABLE 2
Breakdown of GIPSY software on the host computer system disk as of 18 October 1984 by type. The typical packing density for source code is 38 lines per kilobyte and for documentation 25 lines per kilobyte.

File Type		Number of lines
SHELTRAN source code		83,227
FORTRAN source code		13,001
MACRO source code		2,548
	TOTAL	98,776
Documentation	Level 1	16,597
	Level 2	8,205
	Level 3	3,063
	TOTAL	27,865

TABLE 3
Rough Costs of GIPSY in millions of Dutch guilders over 8 years (1977 - 1985).

Hardware items	Cost
PDP 11/70 with peripherals	0.90
I^2S image computer and IPS video disk	0.42
Total Investment (1977)	1.32
Contract maintenance/yr	0.1
Local engineering help/yr (1 day per week)	0.02
Total maintenance costs (8 yrs)	0.96
Total Hardware cost over the lifetime of the project	2.28
Image processing software development costs, 3.5 man-years/year over 8-year lifetime of the project (3.5 × 0.08 × 8). Total software costs	2.24

- simultaneous operation of several application programs is quite useful, and the system provides a good overview of what is going on;
- the use of private data disk packs is much appreciated by the users (it avoids a lot of tape operations and gives them added flexibility in carrying out lengthy data processing jobs), and relieves the system manager from the odious task of allocating and recuperating space on the system disk;
- the software "environment" of interface subroutines eases the job of developing new application programs;
- it is easy to test new programs;
- system maintenance is simplified by the standard updating scheme with automatic backup;
- the standardization on floating point numbers simplifies the software; and
- the adoption of common programming standards has led to an integrated system with a high level of modularity and internal consistency.

3. Present Inadequacies

There are a number of areas for improvement in the future:

- the implementation of the data base structure on disk is not sufficiently flexible. For instance, the astronomical headers are of fixed length and cannot be extended, and subsets cannot be separately deleted from a data set;
- many users are of the opinion that it would be helpful to their bookkeeping to have a history file attached to a data set. This file lists the processing operations carried out on that set in chronological order;
- many application program tasks are so large that they must be built up of overlaid segments in order to fit in the 56 kilobytes of address space available (after subtraction of 8 kilobytes shared region required for inter-task communication). Constructing overlays requires an unnecessarily detailed knowledge of the computer hardware and operating system and is prone to errors;
- the image computer is not fully integrated into the data base structure; and
- the application programs currently available are quite heavily oriented to the analysis of radio astronomical images, both maps and spectral line profiles. More application software is needed for handling optical images, both surface photometry in the map plane and spectra in the profile plane. The software currently available is useful for radio data cubes from the Westerbork telescope and the VLA, for optical emission-line data cubes obtained with TAURUS, and for sky maps obtained with IRAS. GIPSY is not yet able to conveniently handle IPCS spectra, galaxy photographic surface photometry, or radio

single dish line profiles or interferometer visibility data.

Finally, we mention several points occasionally presented as inadequacies of GIPSY, but which we consider debatable:

- the master control program does not allow user access to operating system utility programs;
- the programming language (Sheltran) is not widely known;
- communication between tasks is limited to "Keyword=parameter" pairs. Large data buffers cannot be transferred, and synchronous simultaneous operation of tasks is not supported;
- the system does not support "minimum match", so that task names and keywords must be typed in completely; and
- the "HELP" documentation is limited, so that a new user has to know quite a lot about the system before he can get started. There are not enough "Documents about Documents".

ACKNOWLEDGEMENTS

We owe a great deal to the graduate students and staff members who have used, programmed, and criticized GIPSY over the years. Many good aspects are due to them, and they have helped us understand why many of our own ideas were not as good as we thought. Several staff members have made major contributions to the application program library and deserve special mention: Miller Goss, Ulrich Schwarz, and Seth Shostak. Renzo Sancisi has offered many excellent suggestions for improvements (many of which are unfortunately still pending), and has acted very effectively for quite some time as the "user's friend".

The manuscript and illustrations were prepared with the assistance of Ineke Rouwé, Gineke Alberts, George Comello, Wiebe Haaima, and Geert Tamminga.

The PDP 11/70 computer is provided and maintained by the Groningen University Computer Center. We are grateful to the director and his staff for their continued cooperation.

REFERENCES

Allen, R.J. 1979a, in Image Processing in Astronomy, ed. G. Sedmak, M. Capaccioli, and R.J. Allen (Osservatorio Astronomico di Trieste), 233.

Allen, R.J. 1979b, in Image Formation from Coherence Functions in Astronomy, ed. C. van Schooneveld (Reidel, Dordrecht), 143.

Allen, R.J. 1983, in Three-Day In-Depth Review of the Impact of Specialized Processors in Elementary Particle Physics, ed. Istituto Nazionale de Fisica Nucleare Sezione de Padova (Padova, Italy), 323.

Allen, R.J., Atherton, P.D., Tilanus, R.P.J. 1985, in The Milky Way Galaxy, ed. H. van Woerden et al. (Reidel, Dordrecht), 275.

Allen, R.J., Goss, W.M. 1979, Astron. Astrophys. Supp. 36, 135.

Allen, R.J., Terlouw, J.P. 1981, in Proceedings of the Workshop on IUE Data Reduction, ed. W. Weiss (Observatory of Vienna), 193.

Brooks, F.P. Jr. 1975, The Mythical Man-Month (Addison-Wesley).

Chu, Y.-H. 1982, Astrophys. J. 254, 578.

Croes, G.A., Deckers, F. 1975, Informatie 17 (Netherlands Society for Computer Science, Paulus Potterstraat 40, Amsterdam), 109.

Ekers, R.D., Allen, R.J., Luyten, J.R. 1973, Astron. Astrophys. 27, 77.

Simkin, S.M., Bosma, A., Pickles, A.J., Quinn, G., Warne, D. 1983, Wisconsin Astrophysics Series (Univ. Wisconsin).

SGP/7, Starlink General Paper 7, 1 July 1980 (Rutherford and Appleton Laboratories Computing Division).

SGP/18.1, Starlink General Paper 18.1, 11 September 1981.

Appendix A. Special HERMES commands.

Control character commands are produced when the user types a letter key while pressing CTRL (control):

^A	if only one program is active: abort it;
^D	list data sets in use;
^E	step forward in table of active display programs;
^F	make hard-copy of grey-scale image currently displayed on TV;
^G	make hard-copy of graphics plot currently displayed on TV;
^H	(backspace) same action as DEL;
^I	(TAB) display previous page and enter page mode;
^J	(line feed) display next page;
^K	activate servant display program KIJK;
^L	display highest page number ever displayed before;
^M	(carriage return) terminate input line;
^N	make hard-copy of terminal screen contents;
^P	enter or leave page mode;
^R	rewrite Task Status Area;
^T	activate cursor processor program for plots;
^U	delete text currently present in User Command Area;
^W	step backwards in table of active display programs;
^Y	activate cursor processor for images;
^Z	when typed three times: stop HERMES immediately;
DEL	delete last character in User Command Area;
SPA	(space bar) if User Command Area is empty or only contains a keyword prompt: step forwards in table of programs requesting keyword input;
ESC	step forwards in table of currently active servant programs and display the program name in the User Command Area.

Appendix A. (continued)

Other commands:

name /ABO	abort servant program;
/END	stop HERMES;
/ERRLEV n	set error level to n;
name /GO	resume a suspended servant program;
/HC l:m	make hardcopy of log file pages l through m;
/MESLEV n	set message level to n;
/MODE n	set output mode to n (Ø = experienced, 1 = normal, 2 = test mode);
/OFF dev	switch device logically off;
/ON dev	switch device logically on;
/STAT n	set status message level to n;
name /WAIT	suspend a servant program;
*n	display log file page number n;
*+n	display page number being displayed + n;
*-n	display page number being displayed - n;
=string or +=string	search forward for "string" and, if found, display the page containing it;
-=string	search backwards for "string".

Appendix B. GIPSY applications programs listed by function.

```
*** Ø. GENERAL INFORMATION  ***
       ------------------
       (these are documents which can be displayed with HELP)

HERMES          user's manual
PROGRAMS        list of programs
INDEX             "          "     alfabetical
USERS           list of users, telephone numbers
PDP11           user's guide to PDP11/7Ø
M7ØE              "       "       M7ØE image computer
GIPSY             "       "       GIPSY system
VIDEO             "       "       VIDEO DISK
DISKPAKS        list of allocation of diskpacks to users
JANUS           how to start and end a session
REDLINE         guideline for HI line data processing
INPUT           specifies input format rules for parameters
COMP            overview of programs for handling component files
TVCAM           describes use of TV camera
TRICKS             "       a few tricks
PHOTOS             "       how to make photos from display screen
EDN11           a poor man's guide to the Edinburgh editor EDN


*** 1. UTILITY TASKS   ***
       -------------

NEWS            gives daily news
HELP            displays documentation
EDN             edits text files
FILE            lists all files (except maps) on your disk
REPEAT          defines sequence of tasks for repetitive execution
DLY             allows delay before input
COORD           coordinate transformations
CURSOR (^Y)     gets coordinates from displayed image at cursor position
TXCUR  (^T)       "       "         "   plots displayed on screen
REWIND          rewinds mag tape
END             stop HERMES
BYE             stop HERMES and log off


*** 2. DATA TRANSFER    ***
       -------------

RDFITS          reads FITS tapes onto disk
LFITS           lists   "    "
WFITS           writes  "    "
RDMAPS          transfer Leiden maps from mag tape to disk
MOSAIC          loads display system with maps from disk
LOADRM          transfers map from display system to disk
RECORD          records TV-camera image into display
MTRANS          transfer images within display system
GTRANS            "       graphics plane images
TRANS           transfers maps from set to set on disk
THEAD             "       map headers "     "       "
MOVIMG          transfers images between storage devices
MSAVE           save & restore the complete M7Ø status
```

Appendix B. (continued)

```
*** 3. DISK                    ***
       ----
DISK (^D)        lists disk sets
HEADER           displays a specified map header
FILE             lists all files except maps
MACRO            lists macro files
DELETE           deletes sets and other files from disk
ANTPAT           specifies set and subset for AP, and computes HPBW
FIXHED           changes map headers
MNMX             finds min and max in a map and updates map header
COM              to add or recall comments
KRIMP            extract  subfield from map
INSERT           insert detail map into bigger map
DECIM            reduce mapsize by grid decimation
REGRID           interpolate map to new grid
TRANS            transfers maps from set to set on disk


*** 4. VISUAL DISPLAY   = general  ***
       --------------

LOOK             display of map full-screen
LOOK REC=Y                "      "      "      and records on VIDEO
L                rapid display of single map (enhance only)
KIJK (^K)        enhance + zoom + pseudocolor
MOSAIC           loads display with maps from disk in a mosaic
COLOR            colour display-continuous spectrum
CLEAR            clears display and de-zooms
MHED             header information for images in display
MSAVE            save and restore the complete M7Ø status


*** 5. VISUAL DISPLAY   = special  ***
       --------------

CAL              writes annotated calibration wedge
PLUS             plots positions (crosses) on TV-screen
TITLE            writes titles on displayed maps
TEXT             writes text on screen for FOTO and photos
ENHANS           enhances contrast & brightness of display
ZOOM             zooms image on screen
ROAM             panning and zooming over entire display system
PSEUDO           pseudo-colouring of images
KLEUR            false colouring on 3 images
STEPS            quantitative discrete level display
COLBAL           change M7Ø colour balance
COLINT           interactive combination of colour & intensity
VELCOL           colour display of velocity field
FFTCOL             "      "     " amp.& phase
GAUCOL             "      "     " output GAUSS or WINDOW
REK              stretches displayed map in Y-direction
ENTER            pushes images up M7Ø stack
ROLL             rolls    "   down   "
BLINK            blink between images in display
SEQ              forms sequence of images resident on VIDEO disk
HIST             for turning graphics planes on & off
ZWFOTO           set up display for making black & white photo
PHOTOS           how to make photos from display screen (doc)
```

Appendix B. (continued)

```
*** 6. PLOTTING,PRINTING and  HARD COPIES       ***
      ---------------------------------

HC (^N)          copy of terminal screen pages
CPLOT            contour plot on VERSATEC or TV
FPLOT              "      "      "   with halftone
PPLOT            plot profile on TV (use ^G for VERSATEC copy)
PLVIEW           sends plot(s) to a specific device
RULSRF           ruled surface map on VERSATEC
GRCOP (^G)       VERSATEC copy of CP plot from TV screen
VECTOR           vector plot (for polarization and gauss components)
FOTO (^F)        grey scale copy of image on VERSATEC
TEXT             writes text on screen for FOTO documentation
PRINT            numbers on the line printer or terminal screen
RADPRT           intensity map on the line printer
PROF             plots profiles (constant M) on line printer


*** 7. MAP MANIPULATION         ***
      ----------------

SMOOTH           smooth maps to lower resolution
PRESM            determines parameters of smooth
MCONV            smooth 8-bit display image
HAN2             2-dimensional hanning smoothing
HANVL            hanning smoothing XV,LV or VL-maps
VELSMO           smooth a set of channel maps in velocity
STRIP            strip integration in Y-direction
SCALE            (new map)=A*(old map)+B
MNMX             finds min and max in a map and updates map header
EDIMAP           changes numbers in maps
BLOT             blotches maps(e.g.zeroes outside blotched area)
ZERO             zeroes map or part of map
CONDIT           conditional transfer of maps from set to set
CLIP             transfer maps with values > &< cutoff
RING             corrects for error rings(e.g.DC offsets)
INSERT           insert small into larger map(also corrects for DC-offsets)
LEWIS            180 degree rotation and symmetrize (removes phase information)
MIRROR           mirrors a map left-right or up-down
ELLINT           integrates maps in ellipses
PBCORR           corrects for primary beam attenuation
ROTATE           rotates maps (useful also for interpolation !)
ROT90              "     "    90,180,270 degrees
DECIM            reduce mapsize by specifying decimation factor
REGRID           interpolate; define new grid


*** 8. MAP COMBINATION    ***
      ---------------

MEAN             mean of channel maps
ADD              adds      "     "
SUB              subtracts a map from a set of maps (eg. line-continuum)
CONREM           subtracts interpolated continuum
CMBMEM           combines display images
VEL              makes total HI maps, velocity fields, etc.
DIV              divides a series of maps by a map
MUL              multiplies a series of maps by a map
PAIR             combines pairs of maps (add, subtract, multiply, divide)
TAU              computes optical depth
SPECT            computes spectral index
POL              calculates linear polarization intensities and position angles
```

Appendix B. (continued)

*** 9. MAP ANALYSIS and SOURCES ***

STAT	computes noise statistics and fluxes
FLUX	calculates fluxes
HIST	histogram
RANDOM	puts random numbers into map
FIND	measures parameters of discrete sources
FINDGA	Gauss find for point sources
GAUSAD	adds 2-D Gaussians to maps
SUBTRA	subtracts response of a point source
CLEAN	cleans sources
SCL	fast in-core CLEAN for small area
MCLEAN	image computer clean
FILE	lists components file
DELETE	deletes " "
RESTOR	restores sources on residual map
ESUB	subtracts components extended source

*** 10. PROFILE ANALYSIS ***

LV	L-V maps
STRIP	generates strip integrated L-V maps
XV	position-velocity maps (arbitrary direction)
PPLOT	plot profile
WINDOW	generates profile parameters for set of channel maps
VEL	makes total HI maps, velocity fields, etc.
GAUSS	fits gaussians to line profile
GAUCOL	velocity coded colours of output GAUSS or WINDOW

*** 11. UVDATA and FOURIER ANALYSIS ***

FFT	fast Fourier transform (1 and 2-D)
SLOFT	slow " "
FFTCOL	color display of amplitude and phase
SUBFT	subtraction of sources using the FFT
EDIUV	edits UV data
UVDISP	displays WSRT (u,v) data from mag tape
CURSUV	cursor for UV data

*** 12. GAUSSIAN ANALYSIS ***

GAUSS	fits Gaussians to line profile
GAUEST	initial estimates for gaussian fitting
GAUCOL	color display of output GAUSS or WINDOW
GAUSEL	selects gaussian components
GAUEDI	edits " "
WOLKCR	cross reference of cloud numbers
WOLKST	calculates parameters of clouds
GAUSUM	" sum of gaussians for HI global profile
GAURUL	plots components in ruled surface map
VECTOR	plots components
GAUSAD	adds 2-D gaussian to map
FINDGA	Gauss find for point sources

*** 13. MODEL FITTING and ANALYSIS ***

MAPFIT	generates model channel maps etc.
VELFI	calculates model velocity field
RCURVE	derives rotation curve from velocity field
ELLINT	integrates maps in ellipses
RANDOM	puts random numbers into maps

Appendix C. Example of a level 1 (user) document.

```
Title:          HISTOG

Purpose:        Program to make a histogram of intensities
                in a map or a part of a map.

Author:         Th. Jurriens

Keywords:       SET=        input set and subset.
                            If more than one subset is given,
(** = hidden)               the program will repeat for each
([] = defaults)             subset using the same parameters.

                AREA=       [whole map]

                NLEVS=      the number of levels [100]

                DEV=        output device for plots Gould, TV
                            and/or plot file [TV, file is saved]

                CLIP=       lowest and highest intensities in
                            the histogram (2 values)
                            [minimum, maximum in AREA]

              ** CUM=       plot cumulative values [NO]

              ** PRINT=     print histogram values [NO]

              ** LEVEL=     counts the number of pixels above this
                            level [minimum in area]

              ** CSTEP=     constant bin size?  [NO]

              ** TYPE=      type of histogram plot BAR or NBAR [BAR]

Notes:          1. the program uses ARTIST to make a plot

                2. the program calculates the mean flux in AREA
                   and reports the total number of pixels used
                   in AREA excluding UNDEFINED values.

                3. if CSTEP=NO the bin width is calculated from CLIP
                   and NLEVS; otherwise STEP= is requested and NLEVS
                   is ignored.

                4. the program gives the mean mode and entropy of the
                   histogram.

Updates:        28-nov-83 document proposal created.
                 7-nov-84 update
```

Appendix D. GIPSY inferface routines listed by function.

+ Documents only * Slave task

USER INTERFACE:

ANYCOO/DANYCO	ask for position parameters;
ANYOUT	general message output;
ASSERT	verify logical condition;
BOX	define rectangular area of map;
EASYOU	print contents of unit 1;
KRUIS	regenerate cursor;
REGION *	define irregular blotch region;
SETINP	ask for set, subset;
USRINP	general parameter input routine;
ZBLCH	define irregular blotch region.

DATA BASE INTERFACE:

CHMAP	check if subset is present;
CHSET	check if set is present;
CMPOSE	transfer images in mass storage;
COMB	perform arithmetic operations on maps;
CPHEAD	copy map headers;
GETPOS	get positions in celestial coordinates;
HEADERS +	how to handle headers;
MAPHED	print header information;
MOVIMG *	general image transfer task;
NPOS	number of positions created by PUTPOS;
PRTHED	print astronomical header array;
PUTPOS	store component positions;
READXY/WRITXY	read an area of a map;
RHEAD/DRHEAD	read map header arrays;
RDBHED/WDBHED	direct access to set or subset header;
WHEAD/DWHEAD	write map header arrays.

DISPLAY INTERFACE:

DSPFILE +	interface routines for display device;
DSPHED	print map title on TV display;
FCBERR	group error messages for display system;
MEMASK	return memory channel address for display;
MHEAD	read/write display image headers;
MSCOFF	access parameters in display header;
PLMASK	returns memory bit plane address;
PLOT +	description of plot software;
PLOTSP *	background task for plotting;
PLTINP	read plot header;
RECSSF	record displayed image on video disk;
SETDSP	access to sets in display system;
UNIFRM	draw frame around a plot.

ERROR HANDLING AND STATUS MESSAGES:

ERROR	error handling and reporting;
STATUS	print message in Task Status Area.

MASTER CONTROL PROGRAM (HERMES):

CANCEL	remove keyword from active table;
CONBOX	connect to TV display cursor box;
DEPUTY	start a slave task from another task;
DISCON	disconnect from HERMES;
FINIS	finish running this task;
INIT	inform HERMES about this task;
PASFIN	as FINIS for use in PASCAL servant tasks;
PASINI	as INIT for use in PASCAL servant tasks;
REQLUN	centralize the use of logical unit numbers;
RESOLVE	defines, calls and deletes macros;
SUBST	keyword substitution;
WKEY	write keywords to tasks own list;
XEQ	send a task start command to HERMES;
XEQXIT	as XEQ, and exit.

Appendix E. Example of a level 2 (Programmer) document.

```
Name:           USRINP

Purpose:        User input interface subroutine

Files:          USRINP.DC2, USRINP.SHL

Author:         R.D. Ekers

Call:           CALL USRINP(A,N,TYPE,LENGTH,DEFAULT,KEYWORD,MESSAGE)

Parameters:     A  (I-O)    Array of dimension N.  Usually this is the output
  (I=input )                but it is also used to supply default information
  (O=output)                to the calling program.

                N  (I)      Dimension of A (INTEGER*2);
                            can be 1 if A is a simple variable.

                KEYWORD (I) Keyword used to supply parameter in advance.
                            Used as a prompt in the present implementation.
                            A keyword is a character string containing no
                            more than 10 characters.  The last character
                            must be an equals sign (=).  The other
                            characters must be letters or digits.

                MESSAGE (I) Message to user if keyword is not present.
                            A message is any character string not longer
                            than 55 characters.

                DEFAULT (I) Default level set by calling program (INTEGER*2).
                            Possibilities are:
                            0 = No default is possible, user must answer;
                            1 = Default is input values of A if user replies
                                with "carriage return";
                            2 = Default is input values of A unless user has
                                pre-specified the keyword.

                TYPE    (I) Data type of A desired in calling program
                            (CHARACTER*1). Possiblities are:
                            'F' = real numbers returned unless input contains
                                  alphanumeric information;
                            'I' = Integers are returned;
                            'A' = Character information,
                                  No conversions are made;
                            'L' = Logical.  A is true if reply is yes or ja
                                  or true.  A is false if reply is no or nee
                                  or false.  First letter is sufficient.  Note
                                  that logical variables must be declared as
                                  LOGICAL*1 in the calling program.

                LENGTH  (I) Data type length of A in bytes.
                            See note on on length of logical variables above.

        Array A is filled up with "ENDLIN" double words, "ENDLIN" words,
        or binary zeros depending on TYPE. The number of elements returned in A
        can be discovered with the functions NEL, NEL2 or NCHAR.

Example:        ...
                CUTOFF = 0.0
                CALL USRINP(CUTOFF,1,'F',4,1,'CUT=','Give cutoff [0.0]')
                ...

Related documents:   INPUT.DC1, CANCEL.DC2, NEL2.DC2, NCHAR.DC2

Update history:     ?           Document recreated, W. Zwitser
                  10-Jul-79     HERMES version, J.P. Terlouw
                  19-Oct-81     Small improvements in documentation
                  27-Aug-84     add call CLREF for SNCOM, W. Zwitser
                  15-Oct-84     Improved documentation
```

Appendix F. Sample GIPSY applications program Sheltran source code.

```
VERSION 5.3 (APR. 84)                                   SHELTRAN-77

TARGET
STM.NO

   1        PROGRAM EXAMPL

      *                          COPYRIGHT (c) 1984
      *                  Kapteyn Astronomical Institute
      *    University of Groningen  -  9700 AV Groningen, The Netherlands
      *
      *    This software is furnished under a license for use only on a
      *    single computer system and may be copied only with the
      *    inclusion of the above copyright notice.
      *
      *    This software, or any other copies thereof, may not be provided
      *    or otherwise made available to any other person except for use
      *    on such system and to the one who agrees to these license
      *    terms.
      *
      *    Title to and ownership of the software shall at all times
      *    remain the property of the University of Groningen.
      *
      *    The information in this document is subject to change without
      *    notice and should not be construed as a commitment by the
      *    Kapteyn Astronomical Institute.
      *
      *    The Kapteyn Astronomical Institute assumes no responsibility
      *    for the use or reliability of this software.
      *

      *+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      *
      *     This program is given as an example of how an applications program
      *     should be written in order to be suitable for incorporation into
      *     the Groningen Image Processing System.
      *
      *     The example is the program POL, which calculates maps of the pos-
      *     ition angle and percent polarization from the constituent maps of
      *     the Stokes parameters Q and U.  The percent polarization is:
      *           P = SQRT(Q*Q+U*U)
      *     and the position angle is:
      *           PA = 0.5*ATAN(U/Q)
      *     at each point in the field.
      *
      *     Originally designed by R. D. Ekers, 27-Mar-80
      *
      *     Update history: minor changes 22-May-80 R. J. Allen
      *                     idem 23-Jun-80 RDE
      *                     regenerated 6-Feb-81 RJA
      *     Changed to conform to SOFTRULES, 17-Feb-81, RJA
      *     Changes made to program and task build file, 7-Apr-81, RJA
      *     Changes to conform to Sheltran-77, 20-Sep-84, JPT
```

```
TARGET
STM.NO
      *
      *--------------------------------------------------------------------

   2          PARAMETER (MAXEX=512)                                        maximum line length
   3          PARAMETER (MAXSST=65)                                        maximum number of subsets
   4          PARAMETER (LHED=64)                                          maximum size of map header
   5          PARAMETER (NCHAR=55)                                         maximum size of message buffer

      *         Declaration statements:

   6          LOGICAL POSA, ASSERT
   7          CHARACTER*(NCHAR) CHARS
   8          INTEGER*2 NSU(MAXSST), NSP(MAXSST), NSQ(MAXSST), NSPA(MAXSST)
   9          REAL*4 DAT1(MAXEX), DAT2(MAXEX)
  10          REAL*8 HED(LHED)

  11          CALL INIT                                                    builds link to HERMES
  12          CALL ANYOUT (0,'Polarization parameters, Version 2.1, Sep-84')  tell user who we are

      * request all input sets

  13          REPEAT
  14             CALL SETINP(NU,NSU,MAXSST,NRU,'SETU=','U set, subset [0,1]')  special routine requests list of data sets
  15             CALL PRTHED(NU,NSU(1),0,1)                                 output some standard header information
  16             CALL SETINP(NQ,NSQ,MAXSST,NRQ,'SETQ=','Q set, subset [0,1]')
  17             CALL PRTHED(NQ,NSQ(1),0,1)
  18             CALL SETINP(NP,NSP,MAXSST,NBP,'SETP=',
               -   'Polarization output set, subset [0,1]')
  19             NPA=-1                                                     set default value if user gives CR
  20             NSPA(1)=1
  21             CALL SETINP(NPA,NSPA,MAXSST,NBPA,'SETPA=',
               -   'Position angle output set, subset [no PA calculated]')
  22             IF NPA.EQ.-1

      * position angle not needed

  23             THEN
  23                POSA=.FALSE.
  24                NBPA=NBP
  25                CALL ANYOUT(0,'Position angles are not calculated')      standard output routine for character strings
  26             ELSE
  27                POSA=.TRUE.
  28             CIF
  29          UNTIL ASSERT(NRQ.EQ.NBP.AND.NRU.EQ.NBP.AND.NBP.EQ.NBPA,0,     Pause if condition is not met
               -        'Unequal number of subsets. Correct and /GO')

  30          IF POSA

      * request P cutoff if pos angle is needed
```

```
TARGET
STM.NO

  32          THEN
  32             CUT=Ø                                    makes default CUT=Ø if user gives CR
  33             CALL USRINP(CUT,1,'F',4,1,'CUT=',        standard routine to request input from terminal
                -   'Cutoff in P for calculation of PA [Ø]')   default indicated between [.] brackets
  34          CIF

      * get size of map

  35          CALL BOX(LLO,MLO,LHI,MHI,NU,NSU(1),'AREA=',-1)   special input routine to request an area
  36          LEX=LHI-LLO+1

      * Check map size to be processed:

  37          IF LEX.GT.MAXEX
  38          THEN
  38             CALL ERROR (4,'Map too big for me !')    standard error routine (4=fatal!)
  39          ELSE
  40          CIF

      * loop on subsets

  41          FOR K=1,NBP
  42             NSQK=NSQ(K)
  43             NSUK=NSU(K)
  44             NSPK=NSP(K)
  45             NSPAK=NSPA(K)
                                                          prepare a text buffer
  46             WRITE(CHARS,'('' Computing on subset '',I6,A1)') NSPK,Ø   zero to flag end of characters
  47             CALL STATUS (CHARS)                      put message in Task Status Area
  48             CALL DRHEAD(NU,NSUK,HED,68,1)            standard routine to read image header
  49             HED(3)='P'
  50             CALL DWHEAD(NP,NSPK,HED,68,1)            write image header (opens file if necessary)
  51             IF POSA
  52             THEN
  52                HED(3)='PA'
  53                HED(2)='degrees'
  54                CALL WHEAD(NPA,NSPAK,HED,68,1)
  55             CIF

      * loop on lines

  56             FOR M=MLO,MHI
  57                CALL READXY(NU,NSUK,LLO,M,DAT1,LEX,1,IER)   standard routine to read data from disk
  58                CALL READXY(NQ,NSQK,LLO,M,DAT2,LEX,1,IER)
  59                FOR L=LLO,LHI                         loop on elements
  60                   LL=L-LLO+1
  61                   U=DAT1(LL)
  62                   Q=DAT2(LL)
  63                   P=SQRT(U*U+Q*Q)
  64                   IF P.GE.CUT.AND.POSA
```

```
TARGET
STM.NO

  65                    THEN
  65                       PA=0.5*57.29578*ATAN2(U,Q)
  66                    ELSE
  67                       PA=0
  68                    CIF
  69                    DAT2(LL)=P
  70                    DAT1(LL)=PA
  71                 CFOR
  73                 CALL WRITXY(NP,NSPK,LLO,M,DAT2,LEX,1,IER)                  standard routine to write data to disk
  74                 IF POSA
  75                 THEN
  75                    CALL WRITXY(NPA,NSPAK,LLO,M,DAT1,LEX,1,IER)
  76                 CIF
  77              CFOR
  79           CFOR

  81           WRITE(1,1000) NP,(NSP(K),K=1,NBP)                               unit 1 device for formatted output
  82     1000    FORMAT(' Polarized flux density in set',I3,' subset',
              -  <NBP>I3)
  83           IF POSA
  84           THEN
  84              WRITE(1,1001) CUT,NPA,(NSPA(K),K=1,NBP)
  85     1001   FORMAT(' Position angle for Polz >',F6.2,' is in set',I3,
              -    ' subset',<NBP>I3)
  86           CIF
  87           CALL EASYOU(0)                                                  transfer unit 1 to selected output devices
  88           CALL FINIS                                                      clean up the mess (tell HERMES we are done,
  89           STOP                                                            - close files, etc.)
  90           END

END OF SEGMENT

  OPTIONS IN EFFECT - LINECOUNT=50, LINEWIDTH=131,    SOURCE,    FORTRAN NUMBERING

   0 ERROR(S)             90 TARGET STATEMENTS
```